Einführung in die Objektorientierung

- Klassen vs. Objekte
- Konstruktoren

Klassen vs. Objekte

- **Klasse** beschreibt die Eigenschaften und Fähigkeiten gleichartiger Objekte
- Beispiel: Entwerfen einer Verkehrssimulation
  - Was ist ein Auto - Klasse oder Objekt?
    - Welche Farbe hat ein Auto?
    - Welche Höchstgeschwindigkeit hat es?
    - Wo befindet es sich gerade?

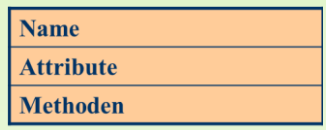
    → Antworten nur für ein bestimmtes Auto möglich
    → „Auto“ bezieht sich auf die *Klasse* Auto

Klassen vs. Objekte

- **Objekt** ist individuelles Exemplar (physisch oder konzeptionell) mit bestimmten
  - **Eigenschaften (Attribute / Datenfelder)**
  - **Fähigkeiten (Methoden)**
- Beispiel: Das Auto daheim in meiner Garage
  - Objekt – spezielles Exemplar eines Autos
    - Das Auto ist rot
    - Fährt nicht schnell
    - Steht in der Garage

Klassen vs. Objekte

- standardisierte Sprache der Modellbildung: **UML** (Unified Modeling Language)
  - wichtigste Diagrammart: Klassendiagramme
  - Darstellung einer Klasse und seiner Beziehungen
- Darstellung von Klassen:



Klassen vs. Objekte

- **Objekt** ist Instanz (konkrete Umsetzung) einer Klasse.
- Eine **Klasse** ist eine gedankliche oder reale Einheit.
- Wozu Objektorientierung?
  - Programmierung: Abbild der Realität
  - Einfache Wiederverwendbarkeit und Wartung

Klassen vs. Objekte

- Klassendefinition
  - Beispiel Klasse Triangle
  - Beispiel Methode changeColor()
  - Eigenschaft / Methode

Klassen vs. Objekte

Allgemeine Klassendefintion:

```
/** Dokumentation */

public class <name>
{

 Datenfelder
 Konstruktoren
 Methoden

}
```

Dokumentation:
/** mehrzeilige Kommentare */
// einzeilige Kommentare

public class <name>: Kopf / Signatur der Klasse

public und **class**: Schlüsselwörter oder reservierte Wörter; (key words)

Rumpf / Body der Klasse zwischen **{** … **}**

< >: Platzhalter für einen Bezeichner (Name des Datenelementes) oder Identifikator

Einführung in die Objektorientierung

- Objekt erzeugen
  - Objekt erzeugen mit Java Code
  - Klasse als Bauplan eines Objekts

Einführung in die Objektorientierung

- Objekt erzeugen

```
private Circle sun;
...
sun = new Circle();
```

- In einem Schritt:

```
Circle sun = new Circle();
```


Einführung in die Objektorientierung

- Klasse definiert Bauplan, abstraktes Schema
- Objekt ist konkretes Exemplar / Realisierung dieses Schemas
- Objekt einer Klasse muss explizit erzeugt werden
- Von einer Klasse beliebig viele Objekte ableitbar

new <Klassenname>( )

- Erzeugt ein einzelnes neues Objekt auf Grundlage einer Klasse

Einführung in die Objektorientierung

new <Klassenname>( )

- Erzeugt ein neues Objekt auf Grundlage einer Klasse
  - *Allokieren* des dafür erforderlichen Speicherplatzes
  - Initialisieren der Datenfelder
- Mehrere Objekte durch mehrfaches Aufrufen von new
- Wert des new-Ausdrucks ist Referenz aufs neue Objekt

sun ➔ Referenz für ein Circle Objekt

Zeigt auf das neue Objekt im Speicher

Code kann das Objekt nur über seine Referenz ansprechen

Konstruktoren

Jede Klasse hat einen oder mehrere Konstruktoren für ihre Objekte.

- Ein Konstruktor heißt exakt wie die Klasse,
- Hat keinen Rückgabewert,auch nicht `void`!
- Kann Parameter für Initialisierung des Zustands des neuen Objekts haben.

```
public class Square
{
/** Create a new square at default position with default color
*/
    public Square()
    {
        size = 30;
        xPosition = 60;
        yPosition = 50;
        color = "red";
        isVisible = false;
    }
...
```

gleiche Benennung wie Klasse!

Konstruktoren

- Ein Konstruktor ist keine Methode eines Objekts, sondern ein Weg, Objekte zu erzeugen.
- Ein (anderes Objekt) kann den Konstruktor verwenden:

```
public class Picture
{
...
     public void draw()
    {
        wall = new Square();
       wall.changeSize(100);
        wall.makeVisible();
     }
...
```

hier entsteht ein Objekt

Konstruktoren

- **Setter – Konstruktor**, Verwendung eines überladenen Konstruktors als Alternative zum Einsatz vieler Setter - Methoden

```
public Kunde neuerKunde(String vorname, String name, String geschlecht,
                        String datum) {
  Kunde k = new Kunde(vorname, name, geschlecht, datum);
  return k;
}
```

Konstruktoren

- Herkömmliche Belegung der Objektattribute durch Setter – Methoden bei Erzeugung neuer Objekte

```
public Kunde neuerKunde(String vorname, String name, String geschlecht,
                        String datum) {
  Kunde k = new Kunde();

  k.setMindestbestellwert(10);
  k.setVorname(vorname);
  k.setName(name);
  k.setGeschlecht(geschlecht);
  k.setGeburtsdatum(datum);

  return k;
}
```


Kopier - Konstruktor

- **Kopier – Konstruktor**, welcher eine Kopie eines Kunden Objekts erzeugen kann

```
public Kunde(Kunde original){
  vorname = original.vorname;
  name = original.name;
  geschlecht = original.geschlecht;
  geburtsdatum = original.geburtsdatum;
  warenkorb = original.warenkorb;
  mindestbestellwert = original.mindestbestellwert;
}
```